Kernelized SVM for regression

Let's consider the standard case in which we have a linear model for regression

$$y = w^T x$$

and a dataset $D = \{(x_n, t_n)\}_{n=1}^N$
We can define an error/loss function

prediction of new instance target in dataset

$$J(w) = \underbrace{\sum_{n=1}^N E(y_n, t_n)}_{\text{measures how much the prediction is distant from the values in the dataset}} + \underbrace{\lambda \|w\|^2}_{\text{regularization term used to deal with overfitting}} \qquad \text{with } y_n = w^T x_n$$

We have different ways to define the error function E. Loss function J is parametric in E.

Consider this error function:

$$E(y_n, t_n) = (y_n - t_n)^2$$

square difference between the prediction of the model and the value in the dataset

Solution of the minimization problem:

$$\hat{w} = (X^T X + \lambda I_N)^{-1} X^T t = X^T \alpha$$

$$\alpha = (X^T X + \lambda I_N)^{-1} t$$

If the loss function is not regularized, $\lambda = 0$

Predictions are made using:

$$y(x; \hat{w}) = \sum_{n=1}^N \alpha_n x_n^T x$$

If we consider the sum of squared errors as the error function, the solution is a linear combination of terms $x_n^T x$ who's coefficients are α_n.
We can apply the Kernel trick since the input appears in the form of cross product and we have the Gram matrix K in α.

Apply the kernel trick:

$$y(x; \hat{w}) = \sum_{n=1}^N \alpha_n k(x_n, x)$$

$$\alpha = (K + \lambda I_N)^{-1} t$$

the linear model is now expressed in terms of the Kernel function

Issue: computation of K requires $|D|^2$ operations and K is not sparse.
computation of K depends on the size of the dataset.

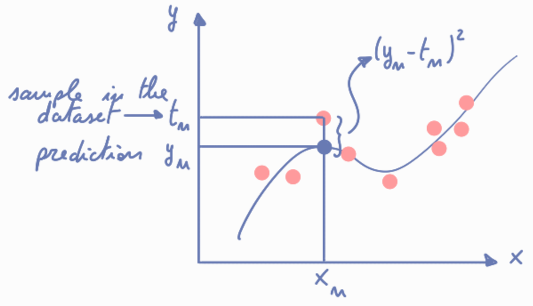
This method may not be practical when the dataset is large

We can consider another formulation of the problem that will not require us to compute k for every possible combination of the input samples. → significant increase in performance

The idea is to use a different error function

Before

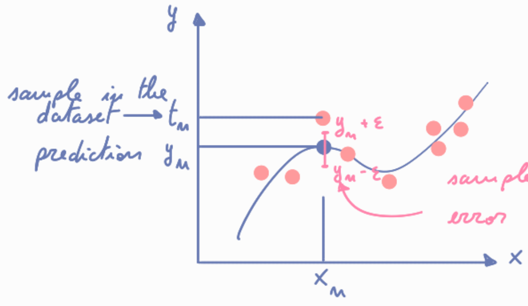
$$E(y_n, t_n) = (y_n - t_n)^2$$



After

this is called ε insensitive error function

$$E_\varepsilon(y_n, t_n) = \begin{cases} 0 & \text{if } |y-t| < \varepsilon \\ |y-t| & \text{otherwise} \end{cases}$$



samples that have an error within this interval will contribute with $E=0$

Samples outside the range will contribute with an error proportional to how far we are from the boundaries

If the error is 0 it will not contribute

Thus, consider this loss function

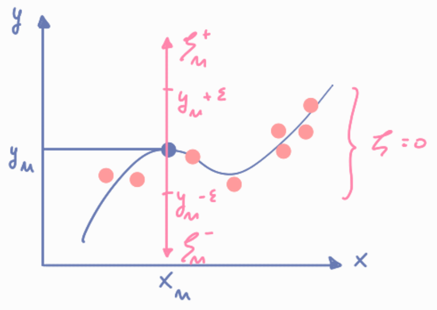
$$J(w) = c \sum_{n=1}^{N} E_\varepsilon(y_n, t_n) + \frac{1}{2}\|w\|^2$$

The problem is that $J(w)$ is not differentiable, thus the minimization problem is difficult to solve.

We need to formulate differently the problem.

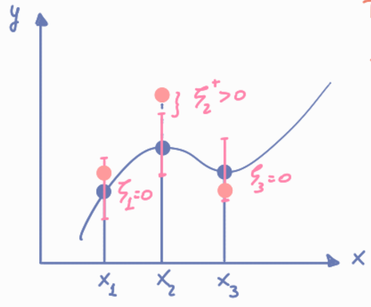
Introduce lack variables $\xi_n^+, \xi_n^- \geq 0$ used to shift the optimization problem on them instead of using directly the parameters of the model.

$$y_n - \varepsilon - \xi_n^- \leq t_n \leq y_n + \varepsilon + \xi_n^+$$



We now need to minimize the value of the slack variables.

Only the samples that are outside the range will have $E>0$ and will contribute.



The error only depends on the slack variables of the points that are outside the ε tube.

$$\boxed{\text{Define a new optimization problem on the } \xi \text{ variables}}$$

This formulation is a standard quadratic programming (QP) problem an can be "easily" solved.

ε is an hyperparameter, you have to do some experimental analysis to find the best one

The greater ε, the more noise you accept

This formulation is better because the Karush-Kuhn-Tucker (KKT) condition holds.
Support vectors contribute to predictions.

$\hat{a}_n > 0 \longrightarrow \varepsilon + \xi_n + y_n - t_n = 0$ data point lies on or above the upper boundary of the ε-tube

$\hat{a}'_n > 0 \longrightarrow \varepsilon + \xi_n - y_n + t_n = 0$ data point lies on or below the lower boundary of the ε-tube

All the data points inside the ε-tube have $\hat{a}_n = 0$ and $\hat{a}'_n = 0$ and thus do not contribute to prediction.
When the lagrangian multipliers are 0 you don't need to compute the Kernel.

When you want to solve a problem with a linear model, the Kernelized SVM with the regularization term is the way to go (the best method you can try).
This under the assumption that the input is a vector of real values $f: R^d \longrightarrow R$